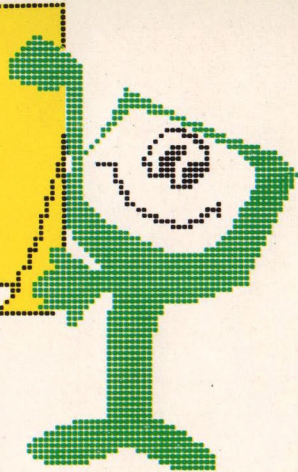
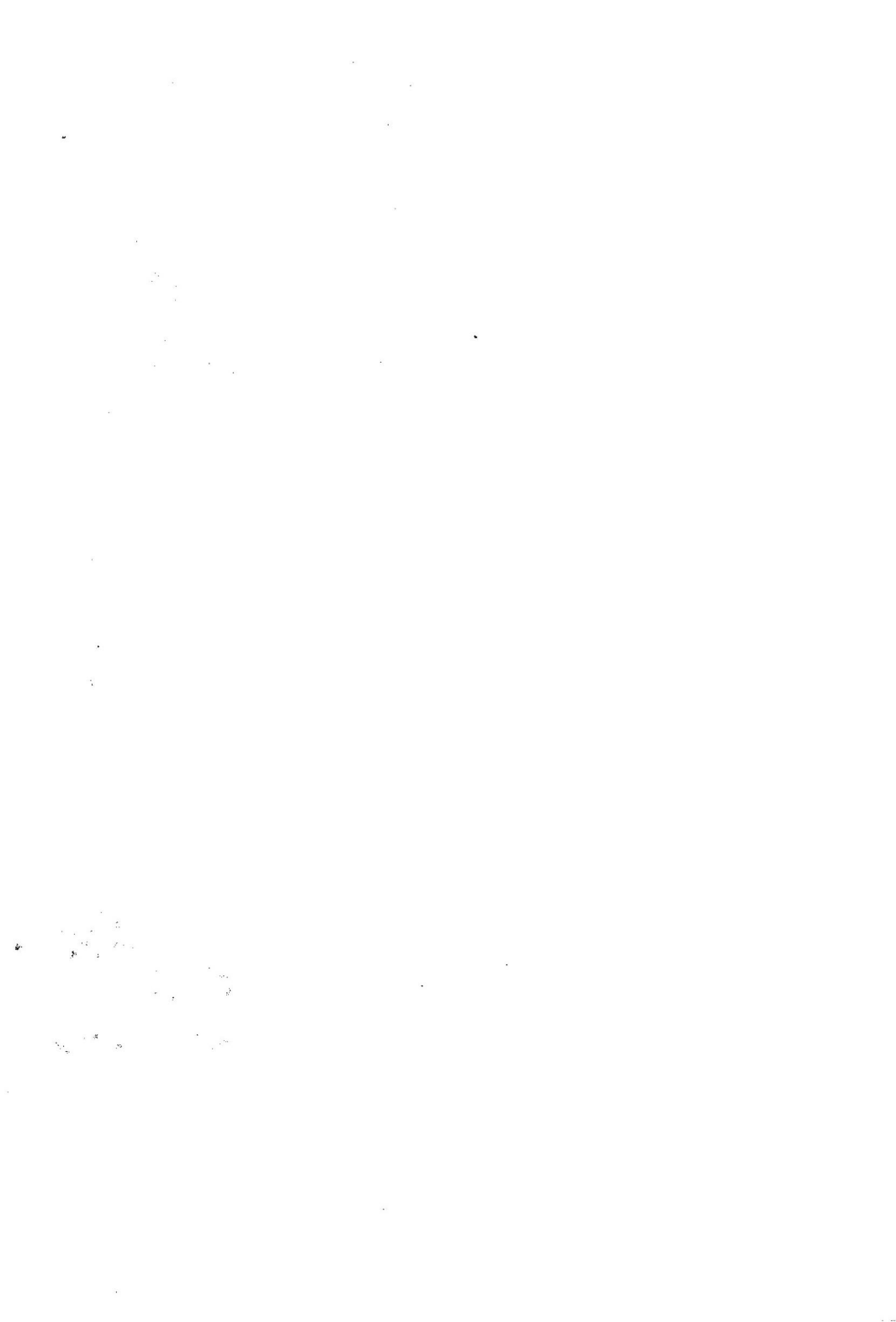
20 VIDEOLEZIONI DI BASIC
PER IMPARARE CON LO SPECTRUM

GRUPPO EDITORIALE JACKSON

Spectrum
16K/48K/PLUS

VIDEO BASIC SPECTRUM
Pubblicazione quattordicinale
edita dal Gruppo Editoriale Jackson
Direttore Responsabile:
Giampietro Zanga
Direttore e Coordinatore Editoriale: Roberto Pancaldi
Autore: Softidea - Via Indipendenza 88 - Como
Redazione software:
Francesco Franceschini, Roberto Rossi,
Alberto Parodi, Luca Valnegri
Segretaria di Redazione:
Marta Menegardo
Progetto grafico:
Studio Nuovaidea - Via Longhi 16 - Milano
Impaginazione:
Silvana Corbelli
Illustrazioni:
Cinzia Ferrari, Silvano Scolari
Fotografie:
Marcello Longhini
Distribuzione: SODIP
Via Zuretti, 12 - Milano
Fotocomposizione: Lineacomp S.r.l.
Via Rosellini, 12 - Milano
Stampa: Grafika '78
Via Trieste, 20 - Pioltello (MI)
Direzione e Redazione:
Via Rosellini, 12 - 20124 Milano
Tel. 02/6880951/5

Autorizzazione alla pubblicazione Tribunale di Milano n° 422 del 22-9-1984
Spedizione in abbonamento postale Gruppo II/70 (autorizzazione della Direzione Provinciale delle PPTT di Milano).
Prezzo del fascicolo L. 8.000
Abbonamento comprensivo di 5 raccoglitori L. 165.000
I versamenti vanno indirizzati a: Gruppo Editoriale Jackson S.r.l. - Via Rosellini, 12 20124 Milano, mediante emissione di assegno bancario o cartolina vaglia oppure utilizzando il c.c.p. n° 11666203.
I numeri arretrati possono essere richiesti direttamente all'editore inviando L. 10.000 cdu. mediante assegno bancario o vaglia postale o francobolli.
Non vengono effettuate spedizioni contrassegno.

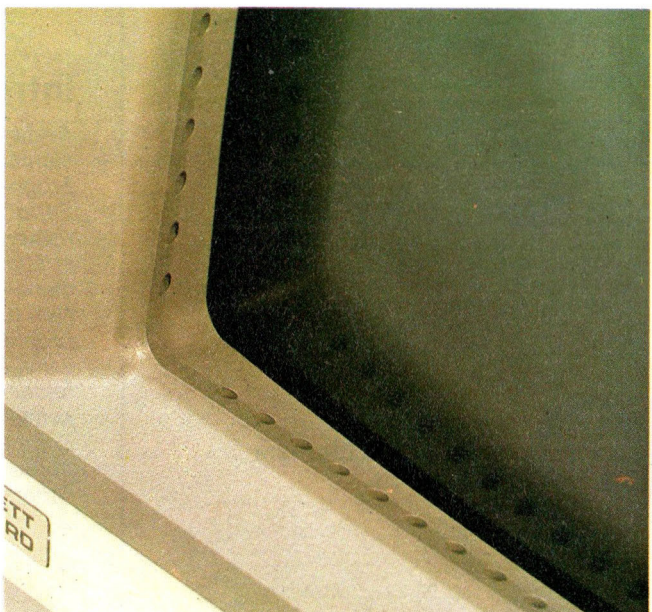
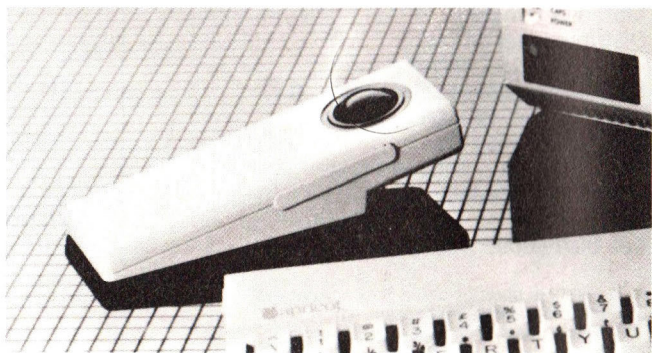
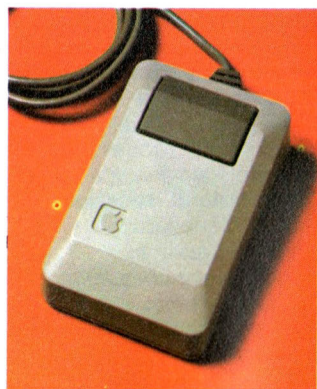
Gruppo Editoriale Jackson

SOMMARIO

Introduzione

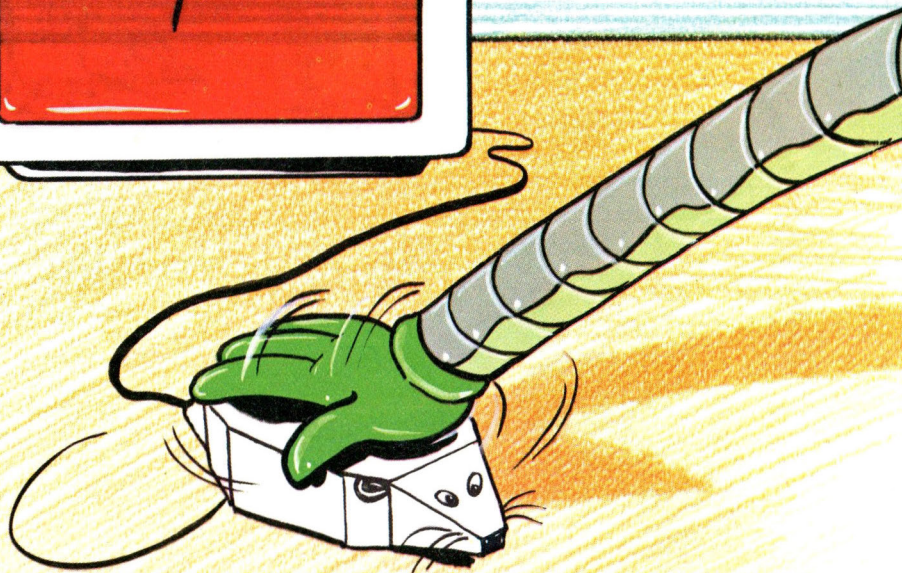
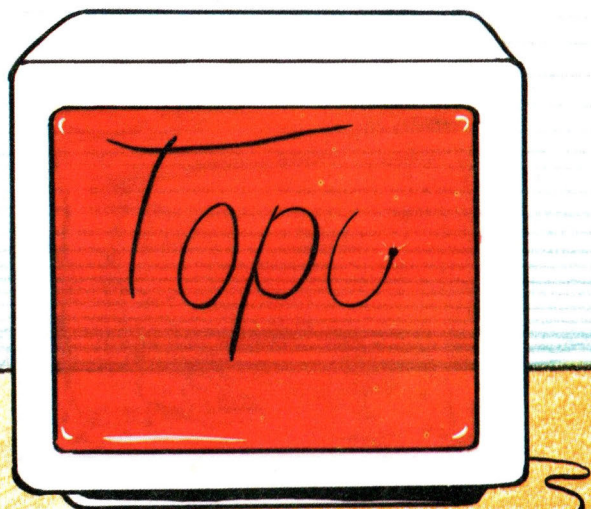
Avete ormai a disposizione tutti gli elementi per essere dei buoni programmatori in BASIC.
Per ottenere dal computer, però, risultati ancora più strabilianti la strada è una sola: il linguaggio macchina.
Non tutto per fortuna deve essere programmato autonomamente; molte volte, anzi, è preferibile in termini di fattibilità, tempi e costi, acquistare software standard.
Ecco allora word processor, apreadsheet, data base e altri "attrezzi" di grande utilità.

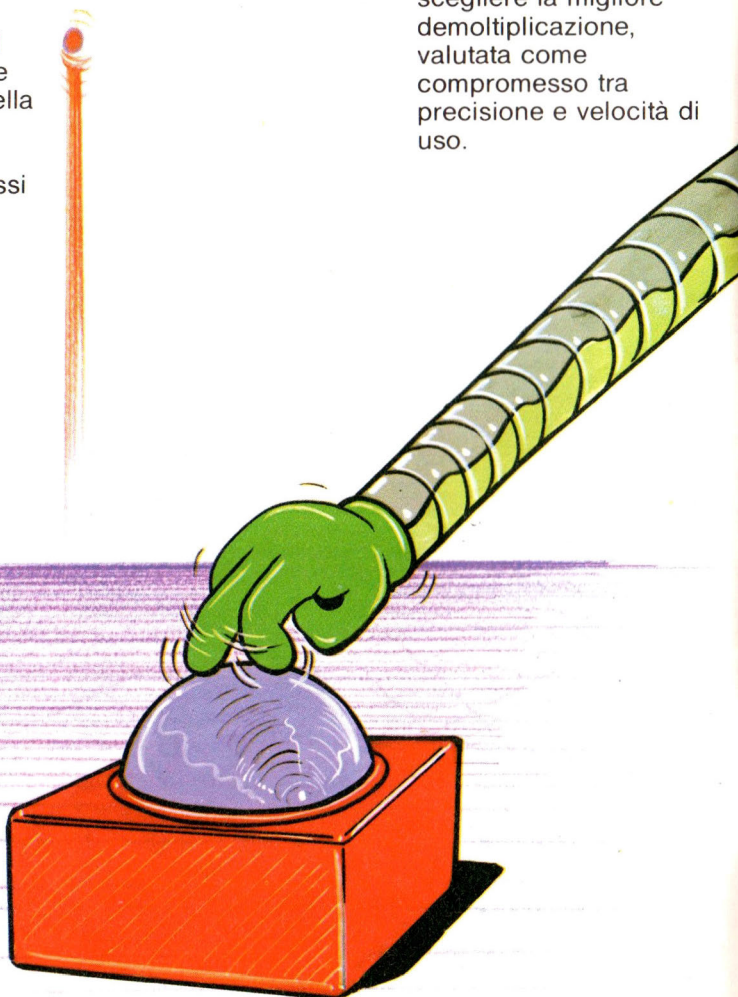
Moderni dispositivi di input

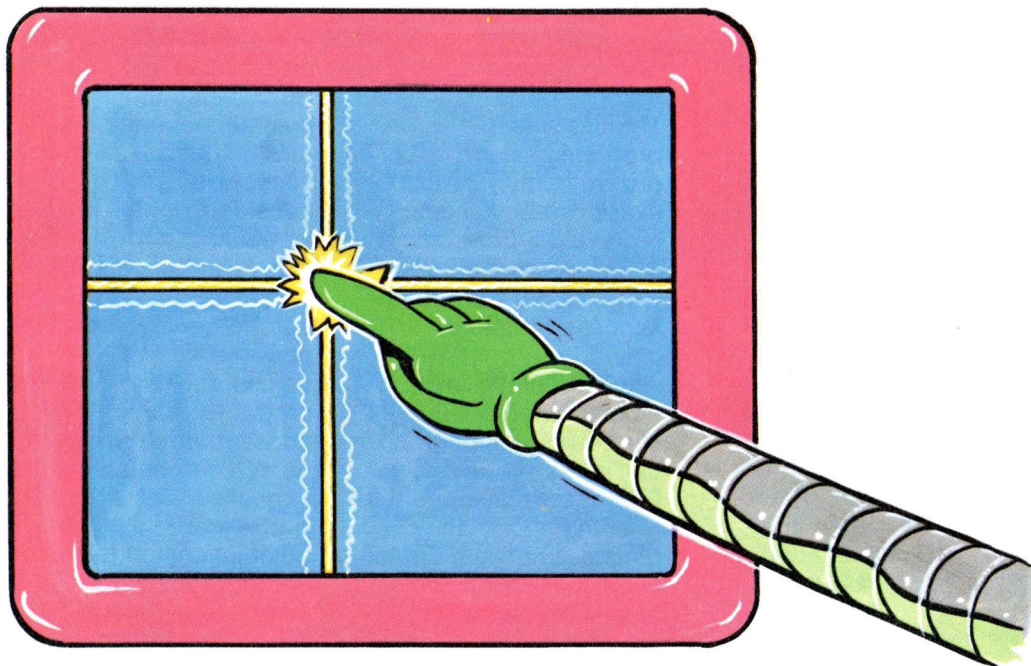
Come abbiamo già avuto modo di vedere, per qualsiasi computer la tastiera costituisce un dispositivo di input estremamente importante, anzi fondamentale.
Esistono tuttavia numerose altre periferiche di input che consentono, di fornire al computer in modo semplice e immediato, i dati che si desidera immettere nella memoria.
Da quando la grafica è entrata prepotentemente nel campo dei personal computer, si sono sviluppati nuovi dispositivi per rendere più semplice ed immediato il colloquio uomo-macchina.
I più diffusi sono il mouse, la trackball e il touch-screen.

MOUSE

Il mouse ("topolino") è della trackball capovolta. La sfera viene infatti posta in rotazione spostando l'intero dispositivo su un piano liscio (per esempio una scrivania).

Rispetto alla trackball, il mouse richiede uno spazio maggiore, essendo necessaria una certa libertà di movimento per la manovra: in compenso l'operazione risulta più intuitiva, in quanto lo spostamento del cursore sullo schermo ricalca fedelmente quello imposto con la mano. È inoltre possibile muovere il cursore tenendo contemporaneamente premuto il pulsante (o i pulsanti) del mouse, cosa impossibile con la trackball.

Per i suoi vantaggi di precisione e facilità di impiego, il mouse si sta attualmente affermando come la principale periferica di input della nuova generazione di personal computer.

TRACKBALL

La trackball (letteralmente "sfera tracciante") è una periferica costituita da una sfera liscia, completamente libera di ruotare in un supporto fisso; viene posta in rotazione utilizzando il palmo della mano. Due rullini posti a fianco della sfera (all'interno del supporto) rilevano la rotazione lungo due assi ortogonali e la convertono in una serie di impulsi elettrici, in numero proporzionale all'angolo di rotazione. Dato che - a differenza di joystick e paddle - non esiste fine corsa, è possibile ottenere tutta la precisione che si desidera, anche se talvolta questo vantaggio va a scapito della velocità di utilizzo.
È pertanto compito del software di "pilotaggio" scegliere la migliore demoltiplicazione, valutata come compromesso tra precisione e velocità di uso.

TOUCH-SCREEN

L'idea del touch-screen è molto simile a quella della penna luminosa: touch-screen significa infatti "video sensibile al tatto".
In realtà non è lo schermo video ad essere sensibile al tocco dell'operatore, ma un foglio trasparente che ricopre lo schermo stesso. Un reticolo di sensori ottici rende sensibile questo foglio al tocco delle dita o di una penna.
Il maggior svantaggio di questa soluzione è costituito dalla scarsissima risoluzione del dispositivo. Inoltre, e questo costituisce uno scoglio notevole, lo strato di grasso sempre presente sulla pelle finisce con l'accumularsi sul video, pregiudicando la lettura dello schermo.
Di contro, l'uso di questa periferica risulta estremamente naturale ed intuitivo, potendo quindi essere utilizzata anche da persone non troppo esperte.

Software di ausilio

Qualunque computer, per quanto sofisticato e perfezionato, da solo non è in grado di operare: occorre infatti abbinare alle caratteristiche hardware della macchina il software appropriato alla risoluzione del problema (o dei problemi) che si intende risolvere.
Per perseguire questo obiettivo, tuttavia, il programmatore deve compiere un insieme di passi ed operazioni variabili di volta in volta.
Occorre quindi analizzare e valutare sotto ogni minimo aspetto il problema, esaminando con attenzione tutte le difficoltà da superare.
Una volta individuato nei termini generali il problema, le varie funzioni da svolgere vengono man mano evidenziate in appositi schemi (comunemente definiti - ormai lo sai benissimo - diagrammi a blocchi), che individuano la successione logica delle azioni, con le possibili situazioni alternative e gli eventuali momenti di decisione. In questa fase si decide anche se l'intero insieme di funzioni da realizzare verrà concentrato in un unico programma oppure suddiviso in più programmi e relativi sottoprogrammi.
Una volta stabiliti i diagrammi a blocchi il programma passa infine alla vera e propria fase di programmazione, realizzata mediante la scrittura delle varie istruzioni in un linguaggio che - come il BASIC - possa essere "capito" dalla macchina.
Terminata la programmazione si provvede quindi (dopo aver preventivamente registrato il programma su un sicuro supporto magnetico a avviare la successiva fase di collaudo.
A questo punto inizia il momento forse più stimolante e impegnativo dell'intero procedimento.
È in questa fase che tutti i nodi vengono al pettine, evidenziando le eventuali lacune che una analisi frettolosa del problema può essersi lasciata alle spalle.
In tale operazione possono sovente essere di aiuto al programmatore quegli insiemi di programmi, comunemente definiti come "software di ausilio", che permettono di ispezionare, valutare e correggere gli errori in modo molto più veloce ed agevole.
I programmi di ausilio più comuni riguardano e permettono operazioni a prima vista molto banali: al momento opportuno essi diventano comunque pressoché indispensabili, specialmente quando si desidera sfruttare in pieno le possibilità del proprio calcolatore.
Ne esistono molti in commercio; anche sulle riviste specializzate essi vengono spesso presi in considerazione. I più completi TOOL-KIT (tool significa letteralmente "utensile", proprio perché questi programmi possono essere considerati gli arnesi del programmatore) consentono di svolgere numerose operazioni, tra le quali:

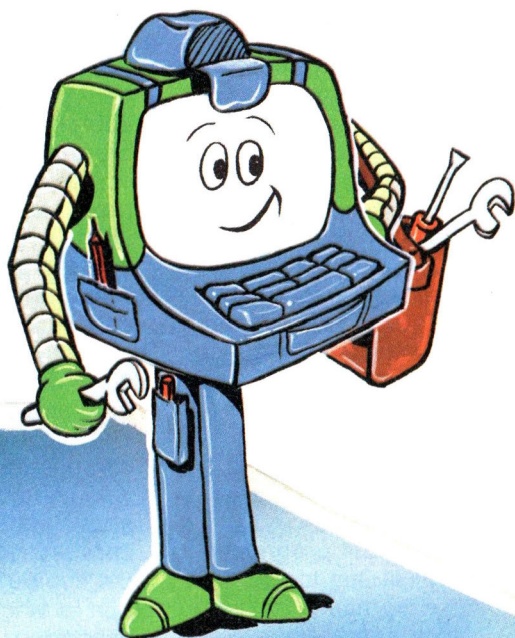
- **numerare** automaticamente le linee del programma mentre si scrive. Si evita così di dover inserire tutte le

volte i numeri di linea; naturalmente, è possibile specificare il passo - cioè l'incremento - che si desidera avere tra una linea e l'altra;
- **rinumerare** un programma che abbia subìto molti aggiustamenti. Questa è una delle possibilità più apprezzate dai programmatori, che spesso - se non esistessero questi programmi - non avrebbero più modo di inserire nuove istruzioni. È chiaro che la rinumerazione delle linee deve implicare - per lo meno in un programma professionale - anche la corretta rinumerazione in corrispondenza dei vari comandi GOTO e GOSUB;
- **cancellare** blocchi di programma. Molti elaboratori permettono di effettuare questa azione: da sistema operativo nel tuo Spectrum è invece possibile cancellare soltanto una linea alla volta. Con un programma di "delete" (cancellazione) è possibile dare comandi del tipo "delete 10-130" (cancella tutte le linee comprese tra la 10 e la 130);
- **ricercare** una particolare stringa di caratteri nel programma. Si evita in questo modo di dover faticosamente ricercare la stringa (per esempio un nome di variabile) per tutta la lunghezza del listato;
- **sostituire** automaticamente una stringa di caratteri con un'altra;
- **visualizzare** i numeri delle linee di programma via via che esse vengono eseguite.

Anche questa possibilità è di estremo interesse e utilità: accade infatti molto spesso che il programma giri correttamente, cioè che fornisca dei risultati in uscita, ma che questi non siano assolutamente corrispondenti a quanto dovrebbe realmente accadere.
Sono questi gli errori più subdoli e difficili da individuare, in quanto non riguardano la sintassi del programma, bensì la sua logica. Con un programma di "trace" (traccia) è allora possibile seguire - attraverso la visualizzazione progressiva dei numeri di linea che vengono eseguiti - lo svolgimento del programma in tempo reale, cioè nel momento stesso in cui esso sta funzionando. Molte volte è sufficiente un'occhiata a questi numeri per essere in grado di risolvere un guaio provocato da un errato salto di riga o da una mancata successione di istruzioni.

Supporti commerciali

Proprio come accade per i capi di abbigliamento, i programmi possono essere su misura o preconfezionati. Questi ultimi, ovviamente, costano meno.
I programmi specificamente progettati e scritti per la soluzione di un determinato problema sono quelli che svolgono il loro compito con la massima velocità, con il migliore sfruttamento delle caratteristiche tecniche del computer utilizzato e con la maggiore rispondenza alle esigenze di chi li deve adoperare.

I vantaggi di questi programmi possono essere riassunti in pochi ma importanti punti:
— sono realizzati professionalmente e collaudati;
— sono pronti immediatamente (non richiedono cioè lunghi tempi di attesa);
— hanno un prezzo abbastanza contenuto.
Analizzando anche il rovescio della medaglia, lo svantaggio principale di questi programmi è che essi non possono soddisfare al cento per cento le esigenze di chi li usa. Esiste infatti sempre un certo margine di "insoddisfazione", dovuto al fatto che un programma progettato per risolvere un problema nel suo complesso non può tener conto di tutti i casi particolari.
Un programma di contabilità - per esempio - non potrà essere identico per un artigiano o per una media industria: per questo molti programmi sono allora "aperti", cioè permettono un margine più o meno ampio di modificabilità, per eventuali "personalizzazioni".
Ci sono invece esigenze che sono realmente comuni ad un numero molto vasto di utenti,

come la scrittura e l'archiviazione dei dati. Infatti i più importanti "pacchetti applicativi" disponibili attualmente sul mercato sono: i word processor, i fogli elettronici e i data base.

Word processor

Un programma per l'elaborazione dei testi (dall'inglese word processor) è una delle possibili applicazioni che, da sola, può addirittura giustificare l'acquisto di un personal computer.
La caratteristica principale di un word processor è infatti quella di trasformare l'elaboratore in una macchina da scrivere molto speciale. Permette infatti di scrivere di getto, quasi alla rinfusa, tutto ciò che si desidera, con la possibilità di modificare qualsiasi parte del testo senza dover riscrivere tutto da capo al momento di battere la bella copia e consentendo di ottenere con la stessa qualità di stampa un numero illimitato di copie. Il suo

uso abolisce quindi in maniera definitiva gomme, correttori, carta carbone e cose simili. Un sistema di trattamento della parola gestisce inoltre automaticamente l'allineamento delle parole sia all'interno delle righe che dei margini, i quali possono essere variati a piacimento con pochi ed elementari comandi. Ma la caratteristica fondamentale di un programma word processor è comunque quella di permettere la correzione dinamica di tutto ciò che si è scritto: si può cioè "tornare indietro" e correggere direttamente sul video, lasciando al computer e alla sua stampante il compito di riscrivere l'intero testo su carta.

Le funzioni più importanti e necessarie di un sistema per il trattamento di testi sono:

- a capo automatico. Non è quindi necessario controllare il fine riga, poiché le parole eccedenti la capienza della stessa vengono automaticamente trasferite all'inizio della riga successsiva;
- possibilità di inserimento di caratteri, parole o righe nel testo già esistente, per correzioni o aggiunte;
- cancellazione di caratteri, parole e frasi, con eliminazione automatica dello spazio resosi disponibile nella riga;
- possibilità di reimpaginare automaticamente il testo, sia perché si è voluto variare il numero di caratteri per riga sia perché si è variata la spaziatura tra le righe;

● possibilità della ricerca automatica di una parola o di un gruppo di parole nell'intero testo;
● possibilità di copiare e trasferire parti di testo in altra parte dello stesso documento.

In programmi maggiormente evoluti esistono altre funzioni, più particolari di quelle appena accennate, ma non per questo di minore utilità: il principio

generale di funzionamento resta comunque sempre lo stesso. Naturalmente, qualsiasi word processor permette di salvare il testo su un supporto magnetico.

Fogli elettronici

Una delle applicazioni che ha avuto maggior fortuna e diffusione nel settore degli home e personal computer è sicuramente il trattamento automatico dei prospetti di tipo tabellare (quelli, per intenderci, organizzati per righe e colonne), cioè il cosiddetto foglio di calcolo elettronico o worksheet (talvolta spreadsheet).

L'utilizzo pratico di un tale sistema è molto ampio e vario: va dalle simulazioni di vendita, in funzione dei livelli di produzione, alla gestione del bilancio personale o familiare, oppure all'esame dell'andamento dei costi, ricavi e utili di un'attività commerciale nel corso dei dodici mesi dell'anno.

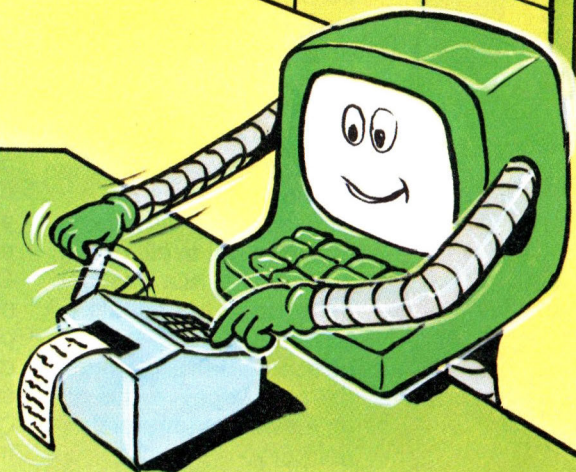
Ma vediamo brevemente come funziona.

Si tratta di un insieme di righe e colonne individuate da un codice (ad esempio, lettere per le colonne e numeri per le righe, proprio come nel gioco della battaglia navale). Ciascuna coordinata può contenere un dato numerico, una formula che collega ed elabora algebricamente o logicamente più dati, o un insieme di lettere qualsiasi, quali quelle componenti un titolo. Tra le diverse informazioni che possono comparire in due o più caselle possono essere definite delle relazioni analitiche, algebriche o di altro tipo (per esempio statistico).

Ciò fatto, è sufficiente introdurre tramite tastiera i vari valori nelle caselle definite come variabili principali, perché tutte le variabili dipendenti vengano automaticamente calcolate dal programma e fatte comparire nelle caselle pertinenti.

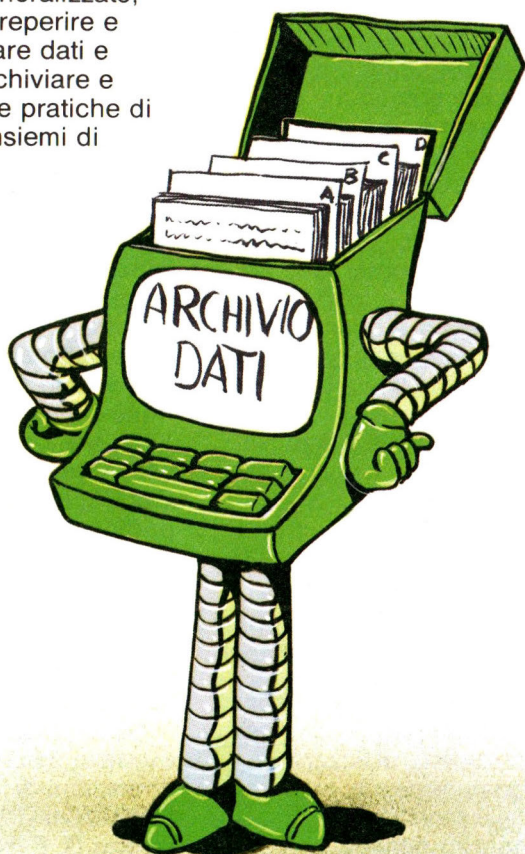
Da questa caratteristica deriva la capacità dei pacchetti di spreadsheet di fungere da potenti strumenti di "simulazione" per modelli di tipo statistico o finanziario. L'acquisto di un foglio di lavoro elettronico è pertanto consigliabile in tutti i casi in cui esista una esigenza,

LINGUAGGIO

GENN.
FEBB.
MARZO
APR.
MAGGIO
GIU.
AFFITTO
CASA
GAS
BENZINA
LUCE
TOTALE

ragionevolmente frequente, di elaborare prospetti tabellari con dati eventualmente tra loro correlati.

Data base

I data base (o, con una libera traduzione italiana, i programmi per l'organizzazione di dati) sono programmi di utilizzo generalizzato, adatti per reperire e memorizzare dati e notizie, archiviare e classificare pratiche di ufficio o insiemi di informazioni e così via. Un data base permette di gestire elettronicamente qualsiasi tipo di archivio di informazioni, disposto e strutturato secondo il desiderio di chi utilizza il programma. Un esempio renderà l'idea nel migliore dei modi. Supponiamo di voler comporre una rubrica telefonica elettronica, contenente - oltre al nome e al cognome (ed ovviamente al numero telefonico) delle varie persone - anche l'indirizzo e la città di residenza.

Con il data base dopo aver introdotto gli elementi è possibile ordinare la lista delle persone, per esempio in ordine alfabetico oppure per zone di residenza, per prefisso telefonico o per qualsiasi altra "chiave".
È inoltre immediatamente ricercabile, in modo assolutamente semplice e automatico, "la persona che ha prefisso X e abitazione con numero civico Y".
Per tornare a un discorso generale, le applicazioni di un data base sono praticamente infinite: tante quanti possono essere gli utilizzi di archivi di informazione. La cosa importante di questi programmi è che essi permettono la gestione di qualsiasi tipo di informazione, non importa quanto strana o complicata, ricercandola ed aggiornandola con la massima rapidità e automazione.

Linguaggio macchina: i registri

Il microprocessore montato sullo Spectrum (ma, più in generale, tutte le CPU) dispone di un certo numero di speciali "locazioni", chiamate registri, che per il programmatore in linguaggio macchina possono essere paragonate alle variabili del programmatore in BASIC.

Sullo Z80 esistono circa 10 registri, ciascuno dei quali viene indicato mediante una lettera dell'alfabeto; tra essi i principali sono:

A, B, C, D, E, H, L

Ogni registro può essere considerato come una particolare locazione della memoria, contenente di conseguenza un valore numerico sempre compreso tra 0 e 255. Esiste comunque (e questo è uno dei punti di forza dello Z80) la possibilità di raggruppare a coppie i registri HL, BC e DE, in modo da poter lavorare con valori anche superiori a questo limite. Naturalmente, questo fatto non limita in nessun caso l'utilizzo indipendente dei registri. Passeremo ora in esame - con maggior attenzione - alcuni tra i diversi registri dello Z80, parlando un po' delle funzioni che li coinvolgono, perché alcuni di loro, e lo vedremo tra breve, sono specializzati, e non li si può certo utilizzare in qualunque circostanza.

— Il registro A. Più spesso denominato come "accumulatore", è il registro più importante del microprocessore. È lui che "accumula" la maggior parte dei compiti nel funzionamento della CPU. In compenso può accettare - a differenza degli altri registri - tutti i modi di indirizzamento disponibili (i modi di indirizzamento ti verranno presentati tra breve). Tutte le operazioni aritmetiche, logiche e di confronto lo usano inoltre come intermediario.

— Il registro B. Questo registro sarebbe relativamente banale, se non fosse utilizzato in modo particolare in una istruzione di salto. Può essere associato con il registro C per formare un registro doppio.

— I registri C, D, E, H, L. Sono semplici registri a 8 bit, che possono essere uniti a coppie: BC, DE, HL. Ad essi si può accedere molto velocemente e vengono in generale utilizzati per conservare i dati. Ognuno di essi è privilegiato riguardo ad alcune operazioni. Il registro C viene per esempio utilizzato nelle operazioni di INPUT/OUTPUT. I registri D, E, H, L vengono invece utilizzati

per il trasferimento di dati.
— Il registro F. Questo registro è il "portabandiera" del microprocessore, nel senso che ciascuno dei suoi bit indica il tipo di risultato conseguente a una certa operazione. I bit del registro di stato (numerati da 0 a 7, 7 quello più a sinistra e 0 quello più a destra) sono specificati con nomi particolari che passiamo brevemente ad esaminare:
.C, FLAG di carry, posizione 0, contiene il bit più significativo dell'ultima operazione eseguita. Il bit di carry assume cioè il riporto generato, nel corso di un'addizione, dalla somma dei due bit più significativi.
.N, FLAG di sottrazione, posizione 1, contiene 1, se l'ultima operazione è stata una sottrazione o un decremento.
.P/O, FLAG di Parità/Overflow, posizione 2, ha un duplice significato. Nelle operazioni aritmetiche indica se c'è stato un superamento di capacità (overflow), invadendo la posizione più a sinistra del byte, usata per il segno. Nelle operazioni logiche indica invece la parità (diventa 1 quando si verifica parità pari).
.AC, FLAG di Carry Ausiliario, posizione 4, viene utilizzato nelle operazioni aritmetiche eseguite su numeri espressi secondo una particolare codifica. È un "mezzo-carry", da cui il suo nome. In effetti dà le stesse indicazioni del bit di carry, ma su un valore di soli 4 bit.
.Z, FLAG di zero, posizione 6, si trova a 1, se l'ultima operazione ha dato risultato zero.
.S, FLAG di Segno, posizione 7, ha il valore del bit più significativo del risultato dell'ultima operazione logica o aritmetica eseguita dal microprocessore.

Tecniche di indirizzamento

Lo Z80 non dispone di una grossa varietà di istruzioni (complessivamente sono 67). Sotto questo aspetto numerose altre CPU risultano nettamente superiori, arrivando facilmente ad avere un numero di istruzioni eseguibili quasi doppio. Ciò nonostante, lo Z80 è uno dei microprocessori più diffusi.
La ragione di questo fatto è molto semplice: tutto è dovuto all'ampia possibilità di tecniche di indirizzamento disponibili che incrementa il numero effettivo delle istruzioni ad oltre 600.
Vediamo innanzitutto cosa intendiamo con l'espressione "tecniche di indirizzamento".
Abbiamo già detto che l'Assembler è un

linguaggio a basso livello, scarsamente (anzi, per nulla) strutturato e dotato di forme logiche veramente minime. In altre parole, l'Assembler si avvale di istruzioni con estrema semplicità operativa ed esecutiva: tutto viene difatti basato sui numeri e sulle locazioni di memoria.
Proprio in virtù di questo fatto, per poter lavorare al meglio in qualsiasi parte della memoria, i costruttori di microprocessori cercano di introdurre la massima flessibilità nelle operazioni eseguibili, fornendo alla CPU le più vaste tecniche di indirizzamento.
L'indirizzamento fa pertanto riferimento al modo in cui deve essere considerato l'operando in una certa istruzione, influenzando di conseguenza anche il risultato della operazione stessa. Esamineremo adesso, uno per uno, i diversi indirizzamenti disponibili, valutandone le possibilità, gli utilizzi, i vantaggi e gli svantaggi.

Indirizzamento diretto su registro

L'operando è un registro. Così

```
LD A, B
```

significa "carica in A il contenuto di B". Le istruzioni di indirizzamento diretto su registro occupano solamente un byte: esse quindi non solo sono brevi, ma consentono pure un'alta velocità. Il tempo necessario per la loro esecuzione è infatti limitato a soli 4 cicli di clock, che nello Spectrum corrispondono a meno di un milionesimo di secondo.
Nota: per ciclo di clock si intende il tempo necessario alla CPU per svolgere un'operazione elementare. Tale tempo viene scandito inflessibilmente da un oscillatore (una specie di orologio al quarzo in miniatura). Tieni presente che - a titolo di informazione - lo Z80 esegue più di 4 milioni di operazioni elementari in un secondo: e questo per tutto il tempo che tieni acceso il computer!

Indirizzamento implicito

Il codice operativo "implica" l'indirizzo dell'operando o di uno degli operandi. Per esempio: RRA serve per eseguire una operazione di rotazione verso destra di un bit del registro A. BIT 0, A esamina invece il bit di posizione 0 del registro A.

Indirizzamento immediato

Questa forma di indirizzamento è utilizzata per caricare con uno specifico valore l'accumulatore o qualche altro registro. Tutte le istruzioni nel modo a indirizzamento immediato sono perciò lunghe due byte. Il primo contiene il codice operativo e il secondo contiene la costante numerica da caricare nel registro o da utilizzare per una istruzione aritmetica o logica. Per esempio, se noi volessimo caricare

nell'accumulatore il valore 160 (esadecimale A0) in modo immediato, potremmo scrivere:

```
LD A, A0H
```

Questa istruzione dice: "carica nell'accumulatore il valore esadecimale A0".

Indirizzamento diretto

L'indirizzamento diretto è il modo in cui i dati sono normalmente recuperati dalla memoria. È specificato dal codice operativo, seguito da un indirizzo a 16 bit racchiuso tra parentesi. L'indirizzamento diretto richiede pertanto tre byte. Un esempio di indirizzamento diretto è:

```
LD A, (1234H)
```

Questa istruzione specifica che il contenuto della locazione di memoria numero 1234 deve essere memorizzato nell'accumulatore.
Lo svantaggio dell'indirizzamento diretto è di richiedere un'istruzione di tre byte: esso è quindi un modo abbastanza lento.

Indirizzamento indiretto tramite registri

Per migliorare l'efficienza dell'indirizzamento diretto viene allora reso disponibile un altro modo, quello indiretto tramite registri.

Con questo indirizzamento una coppia di registri contiene l'indirizzo nel quale si trova il valore da elaborare, cioè l'operando; la coppia di registri viene indicata tra parentesi. In gergo informatico si dice che i registri "puntano" la locazione di memoria.
Un esempio di questo modo è:

```
LD A, (HL)
```

che significa "carica in A il contenuto del byte il cui indirizzo si trova in HL".
L'indirizzamento indiretto tramite registri è nettamente più veloce di quello indiretto, in quanto la CPU non deve leggere l'indirizzo di memoria, visto che esso si trova già nei registri. Naturalmente, occorre caricare nella coppia di registri l'indirizzo desiderato, e quindi questo metodo diventa vantaggioso solo se l'indirizzo viene utilizzato più volte.

Indirizzamento relativo

Questo metodo di indirizzamento è spesso utilizzato per le istruzioni di salto (cioè per i comandi che trasferiscono l'esecuzione da un punto all'altro del programma). Per definizione l'indirizzamento relativo utilizza due byte: il primo è un'istruzione di salto, mentre il secondo specifica lo spostamento e il suo segno. Cosa significa? Supponiamo per esempio di voler mandare l'esecuzione da un certo punto del programma a un altro. Per fare questo dobbiamo specificare (oltre naturalmente all'istruzione di salto) anche il numero di locazioni di cui vogliamo spostarci. Poiché lo spostamento può essere sia positivo che negativo (cioè può avvenire sia in avanti che indietro), l'istruzione di salto - che al massimo deve interessare 255 locazioni - può essere al limite negativa di 128 (salto all'indietro di 128 locazioni), oppure positiva di 127 (salto in avanti di 127 locazioni). Naturalmente, il più grosso svantaggio di questo modo è che non permette salti superiori alle 128 locazioni. Tuttavia, poiché la maggior parte dei cicli tende a essere breve, la diramazione può essere utilizzata quasi sempre e aiuta a risolvere il problema.

Indirizzamento indicizzato

L'indirizzamento indicizzato è una tecnica particolarmente pratica per accedere, uno dopo l'altro, ai dati contenuti in un gruppo di locazioni. Il principio è che l'indirizzo desiderato viene calcolato sommando all'indirizzo contenuto in uno dei registri indice (i registri indice IX e IY sono altre due coppie di registri dello Z80) il valore dell'operando. Così

```
LD E, (IX+5)
```

carica nel registro E il contenuto del byte avente l'indirizzo formato sommando il numero 5 al contenuto della coppia di registri IX. Tutti i modi di indirizzamento (per semplicità ne abbiamo tralasciati alcuni) richiedono comunque, per essere veramente capiti ed utilizzati al meglio, parecchio tempo e (soprattutto) molto lavoro di programmazione. Gli esempi che vedremo tra breve illustreranno qualche possibile applicazione di queste tecniche.

Incremento/ decremento

La più semplice espressione aritmetica che lo Z80 è in grado di eseguire è l'operazione algebrica di addizione e sottrazione del valore 1 dal valore contenuto in un registro. A prima vista questa possibilità non sembra aprire orizzonti sconfinati: in realtà la disponibilità di una simile operazione permette la costruzione di strutture ben più complesse, come per esempio i cicli FOR...NEXT del BASIC. L'importanza dei comandi di addizione e sottrazione è talmente grande che esistono addirittura due istruzioni specifiche per portare a termine questi compiti. Di ciascuna di queste esistono due diverse forme, disponibili con differenti modi di indirizzamento:

ADD istruzione di somma senza riporto
ADC istruzione di somma con riporto
SUB istruzione di sottrazione senza riporto
SBC istruzione di sottrazione con riporto

Tutte queste istruzioni possono influenzare alcuni bit del registro di stato, precisamente il bit di segno negativo e il bit zero.

Il bit di segno viene settato, se il bit più significativo del registro (cioè quello della posizione 7) viene posto a 1 a seguito di incremento o decremento, altrimenti vale zero.
Il FLAG di zero viene invece settato soltanto se, a seguito di una operazione, il registro chiamato in causa contiene il valore zero. Per esempio, incrementando di 1 il valore FFH (255 esadecimale) contenuto in un certo registro, si porterà a 00H il valore del registro, verrà posto a 1 il FLAG di zero (Z=1) e si annullerà il FLAG di segno negativo. D'altro canto, decrementando un registro contenente 00H, si porterà il valore del registro a FFH, ponendo contemporaneamente a 1 il FLAG di segno e a zero il FLAG di zero.

I cicli

I cicli sono una parte importantissima - anzi, vitale - in qualsiasi linguaggio di programmazione: in Assembler lo sono ancor di più, poiché molte istruzioni, che in linguaggio ad alto livello richiedono un unico comando, per la loro esecuzione, in codice macchina necessitano invece di sequenze e ripetizioni più o meno lunghe.

Prima di passare alla dimostrazione pratica dell'uso dei cicli in Assembler occorre però introdurre altri due gruppi di istruzioni: quelle di confronto e quelle di salto.

I confronti

È possibile confrontare il valore contenuto nell'accumulatore dello Z80 con il contenuto di una locazione della memoria (oppure con un valore numerico qualsiasi) mediante le istruzioni

CP confronta il valore
CPI confronta il valore con incremento
CPD confronta il valore con decremento

Il risultato del confronto altera i FLAG di zero (Z), di segno (N) e di riporto, o carry, (C) del registro di stato. In che modo? La risposta è in questa breve tabella:

ACCUMULATORE	CARRY	ZERO	SEGNO
minore del dato	1	0	1
uguale al dato	0	1	1
maggiore del dato	0	0	1

Confrontando per esempio il valore dell'accumulatore (supponiamo che sia 5AH) con il numero B3H, dalla tabella ricaviamo che il valore dei vari FLAG sarà:

C=1 Z=0 N=1

Sulla base di questi risultati potremo quindi prendere le decisioni più appropriate.

I salti

Una volta effettuata l'operazione di confronto può essere necessario eseguire anche un'operazione di salto (così come in BASIC scriviamo per esempio IF A>B THEN GO TO 300).
Possiamo distinguere tre gruppi di salti:

— i salti propriamente detti
— le chiamate ai sottoprogrammi
— i ritorni dai sottoprogrammi.

MA anche due modi di salto:

— assoluti
— relativi.

Un'istruzione di questo tipo:

```
JP ADR
```

si chiama salto incondizionato: essa provoca infatti il salto sistematico all'indirizzo ADR (indicando con ADR un generico indirizzo della memoria). È l'equivalente in linguaggio macchina del GO TO che si usa in BASIC. Queste altre istruzioni sono invece esempi di salti condizionati:

```
JP C,ADR        JP NC,ADR
JP Z,ADR        JP NZ,ADR
JP M,ADR        JP P,ADR
JP PE,ADR       JP PO,ADR
```

"Salto condizionato" significa che il salto deve essere effettuato solo se la condizione richiesta è soddisfatta. Naturalmente, le condizioni di salto condizionato si possono presentare in un programma soltanto dopo istruzioni che posizionano in qualche modo i FLAG. La condizione di salto viene infatti controllata proprio in base ai valori assunti da qualcuno di questi FLAG.

```
....
CP 8            ; confronta A con 8
JP Z, ADR1      ; salta se A=8
JP C, ADR2      ; salta se A<8
                ; allora A>8
.....
```

In questo esempio l'istruzione CP posiziona i flag Z e C. Il salto condizionato JPZ

controlla la presenza del flag Z e provoca il salto all'indirizzo ADR1, se Z è a 1, segnalando in questo modo che il contenuto dell'accumulatore è uguale a 8. In caso contrario il programma prosegue ed incontra l'istruzione JPC, che controlla lo stato del flag C. Se questo è a 1, si effettua un salto all'indirizzo ADR2, altrimenti possiamo essere sicuri che il contenuto di A è maggiore di 8, poiché non soddisfa i test precedenti, e il programma prosegue in sequenza.
Finora abbiamo visto come operando dell'istruzione JP un indirizzo di salto. È però possibile effettuare un salto non più soltanto a un indirizzo dato, ma anche ad un indirizzo calcolato, contenuto in uno dei registri HL, IX o IY:

```
JP (HL)
JP (IX)
JP (IY)
```

Questa modalità di salto torna molto utile in numerose occasioni.
I comandi di SALTO RELATIVO si distinguono dagli altri per una caratteristica speciale: specificano uno spostamento relativo, cioè in avanti o indietro di +127 locazioni o di −128 byte, rispetto alla posizione occupata in quel momento. Le istruzioni di salto relativo sono:

```
JR      VAL
JR C,   VAL      JR NC, VAL
JR Z,   VAL      JR NZ, VAL
```

VAL rappresenta un valore numerico che indica - in modo relativo - l'indirizzo a cui saltare. Per esempio:

```
JR Z, − 14
```

significa: "se il risultato di confronto (che sarà già stato operato in precedenza) ha posto il flag Z a 1, allora esegui un salto all'indietro di 14 byte".
Oltre alle istruzioni di salto vero e proprio, esiste un ultimo comando di salto: CALL. La CALL dell'Assembler è equivalente al GOSUB del BASIC.
Questa istruzione effettua cioè un salto al sottoprogramma il cui indirizzo è specificato nell'operando, non senza aver prima effettuato un salvataggio nell'area di stack dell'indirizzo di ritorno. È quest'ultimo punto che differenzia la CALL da una semplice JP, con la quale le analogie sono peraltro notevolissime.
Per poter tornare dai sottoprogrammi in linguaggio macchina, così come in BASIC è necessario ricorrere a RETURN, bisogna invece impartire un'istruzione RET. In questo caso il microprocessore estrae

un valore dall'area di stack e riporta l'esecuzione del programma all'indirizzo corrispondente al valore prelevato.
Terminata finalmente la parte teorica passiamo adesso a quella pratica, cioè all'applicazione dei concetti visti finora.
Cominciamo subito con un esempio facile, facile.....: supponiamo di voler scrivere un carattere (per esempio una "A") nell'angolo in alto a sinistra dello schermo. Sappiamo già come farlo in BASIC: basta una PRINT AT e il gioco è fatto. Adesso dobbiamo però dimenticarci di queste comodità e cercare di fare tutto con i nostri mezzi. Bisogna innanzitutto ricordarsi il meccanismo per cui i caratteri possono apparire sullo schermo: ciascuna posizione dello schermo è infatti composta da una "griglia" di 8x8 quadratini. Ogni gruppo di 8 quadratini disposti su una stessa riga corrisponde a un byte; ciascun carattere viene descritto compiutamente da 8 byte. Per visualizzare sullo schermo un certo carattere basterà quindi memorizzare nelle 8 locazioni corrispondenti alla posizione prescelta gli 8 valori che lo identificano. Dal momento che la memoria video comincia alla locazione 16384, per visualizzare la nostra A in BASIC potremo fare

```
POKE 16384,0
POKE 16640,60
POKE 16896,66
POKE 17152,66
POKE 17408,126
POKE 17664,66
POKE 17920,66
POKE 18176,0
```

Eseguendo queste istruzioni in modo immediato, vedremo apparire, un pezzetto alla volta, il nostro carattere A. Esaminiamo adesso come fare in linguaggio macchina. Il procedimento non cambierà di molto: bisognerà caricare in memoria i vari valori:

```
LD A,0
LD (4000H),A
LD A,60
LD (4100H),A
LD A,66
LD (4200H),A
LD (4300H),A
LD A,126
LD (4400H),A
LD A,66
LD (4500H),A
LD (4600H),A
LD A,0
LD (4700H),A
```

Carichiamo il linguaggio macchina utilizzando il solito programma BASIC:

```
10 CLS:RESTORE:CLEAR 32500
20 LET INIZIO=32500:LET FINE=33000
30 FOR X=INIZIO TO FINE
40 READ A
50 IF A=999 THEN GO TO 80
60 POKE X,A
70 NEXT X
80 RANDOMIZE USR 32500:STOP
90 DATA....qui vanno scritti i codici
```

Per inserire i codici del programma in linguaggio macchina nella linea

DATA dobbiamo eseguire la conversione delle istruzioni mnemoniche nei corrispondenti valori numerici:

ASSEMBLER	CODICE ESADECIMALE	CODICE DECIMALE
LD A,0H	3E,0	62,0
LD (4000H),A	32,00,40	50,0,64
LD A,3CH	3E,3C	62,60
LD (4100H)	32,00,41	50,0,65
LD A,42H	3E,42	62,66
LD (4200H),A	32,00,42	50,0,66
LD (4300H),A	32,00,43	50,0,67
LD A,7EH	3E,7E	62,126
LD (4400H),A	32,00,44	50,0,68
LD A,42H	3E,42	62,66
LD (4500H),A	32,00,45	50,0,69
LD (4600H),A	32,00,46	50,0,70
LD A,0H	3E,0	62,0
LD (4700H),A	32,00,47	50,0,71

ricordando che il nostro programma BASIC, per non essere interrotto da un messaggio di errore, richiede che l'ultimo valore sia il numero 999. Eseguendo il programma, vedrai comparire sullo schermo - analogamente a prima - la lettera "A" nella posizione corrispondente alle 8 locazioni interessate dalla routine in linguaggio macchina. L'esempio che abbiamo appena visto è abbastanza interessante,

ma non molto pratico: la lunghezza della routine in Assembler sembra eccessiva per un compito così banale. Allora perché specificare la forma del carattere "A", visto che nella propria memoria ROM lo Spectrum già ne conosce benissimo tutte le caratteristiche? Proviamo quindi a ricorrere a una seconda soluzione. Per stampare il carattere sullo schermo dobbiamo conoscere due cose:
1) l'indirizzo dove sono memorizzati gli 8 numeri propri del carattere che desideriamo stampare
2) l'indirizzo (o, meglio, gli indirizzi) che corrispondono a una certa posizione dello schermo.
Nel caso della lettera A, e della prima posizione in alto a sinistra, questi indirizzi sono rispettivamente il 15880 e il 16384.
Per stampare il carattere basterà semplicemente prelevare per 8 volte un byte dalla "memoria caratteri" e depositarlo in una particolare locazione della "memoria di schermo".
Ecco come potrebbe essere una possibile soluzione:

```
LD HL,3E08H
LD BC,4000H
LD A,8H
PUSH AF
LD A,(HL)
LD (BC),A
INC HL
INC B
POP AF
SUB 01
JR NZ, -10
RET
```

Il programma inizia memorizzando nelle due coppie di registri HL e BC gli indirizzi della memoria caratteri e della memoria di schermo. L'accumulatore viene inoltre caricato con il valore 8 (LD A,8A). A questo punto inizia il ciclo vero e proprio.
1) PUSH AF serve per memorizzare in una zona sicura del microprocessore (chiamata area di stack) il valore dell'accumulatore.
2) Si carica il registro A con il valore contenuto all'indirizzo HL.
L'istruzione LD A,(HL) è un esempio di indirizzamento indiretto tramite registri.
3) Si memorizza A nella celletta puntata dai registri BC

```
LD (BC),A
```

4) Si incrementano gli indirizzi puntati da HL e BC.

5) Si recupera (POP) dall'area di stack il valore dell'accumulatore memorizzato in precedenza.

6) Se sottraendo 1 all'accumulatore non si è ancora arrivati a zero, allora si ricomincia il ciclo. L'istruzione JR NZ,-10 utilizza il salto relativo condizionato per terminare il ciclo. La diramazione viene infatti effettuata quando i FLAG N e Z sono settati.
Come puoi notare, questa routine è molto più breve della precedente e, tutto sommato, anche molto più elegante.
I codici numerici che si ottengono dalla conversione delle istruzioni in codice mnemonico sono:

```
33, 8, 62, 1, 0, 64, 62, 8, 245, 126, 2, 35, 4, 241, 214, 1, 32, 246, 201.
```

Tool di programmazione

Visto che ormai stiamo addentrandoci sempre di più all'interno dello Spectrum, possiamo cominciare ad esaminare alcuni "programmi d'utilità", trucchetti e brevi routine, che possono sempre essere di aiuto in particolari circostanze.
Vediamo innanzitutto cosa fare per rendere impossibile la cancellazione della prima linea di un programma: questo accorgimento - apparentemente senza alcun senso pratico - diventa molto utile per tutti coloro i quali desiderano che nella prima riga dei propri programmi appaia una istruzione REM contenente il nome dell'autore o qualche altro avviso.
Il segreto è molto semplice: basta eseguire - dopo aver impostato la linea - la seguente istruzione in modo immediato:

```
POKE 1+PEEK 23635+256*PEEK 23636,0
```

Questo accorgimento darà alla linea il numero 0, in modo da non poterla più cancellare o cambiare. Il processo che consente questa operazione deriva dal fatto che nelle locazioni 23635 e 23636 viene custodito l'indirizzo del programmma BASIC.
Con un minimo di pratica (o, meglio, di sperimentazione) scoprirai che modificando con un altro valore lo 0 della POKE, si possono ottenere altri effetti, diversi da quello che abbiamo visto, ma altrettanto interessanti.
Vediamo adesso un utile programma, che permette di convertire un qualsiasi numero, scritto con una certa base (A), nello stesso numero avente una base diversa (B). Per il modo in cui è stato scritto il programma la massima base che si può manipolare è 36. Le cifre maggiori di 9 sono rappresentate da lettere dell'alfabeto (sia maiuscole che minuscole, in quanto il programma non fa alcuna distinzione): quindi, per esempio, le 12 cifre di un sistema a base 12 saranno:

```
0 1 2 3 4 5 6 7 8 9 A B.
```

Ecco il listato:

```
10 REM CONVERSIONE DI BASI
20 INPUT "DA QUALE BASE?"; A
30 INPUT "A QUALE BASE?"; B
40 PRINT "DA BASE :"; A, "A BASE:"; B
50 INPUT "CHE NUMERO VUOI CONVERTIRE?"
   LINE A$
200 REM DA BASE A A BASE 10
210 LET S=0
220 FOR C=LEN(A$) TO 1 STEP - 1
230 LET D=CODE(A$(C))-48-(7 AND A$(C)>=
    "A")-(32 AND A$(C)>= "a")
240 LET S=S+D*A↑(LEN A$-C)
250 NEXT C
300 REM DA BASE 10 A BASE B
310 LET B$ ="":PRINT
320 LET R=INT(S-B*INT(S/B)+.5):LET S=INT(S/B)
330 LET B$ =CHR$(R+48+(7 AND R>9))+B$
340 IF S < > 0 THEN GO TO 320
350 PRINT A$; TAB 12; CHR$ 61; TAB 16; B$; TAB 9
360 GO TO 10
```

Il cuore del programma è posto nelle linee 230-240 e 320-330. Le espressioni matematiche che puoi leggere in queste istruzioni derivano dal modo in cui lo Spectrum codifica i caratteri (infatti il numero da convertire viene letto come stringa di caratteri, non come numero). Puoi infine osservare che il passaggio dalla base A alla base B avviene mediante la trasformazione intermedia del numero dalla base A alla base 10.
Esaminiamo adesso un altro programma, di genere completamente diverso da quello appena visto, ma non per questo meno utile: è infatti una routine di renumber. Come già ti è noto, un programma di renumber è di grossa utilità nella programmazione, per riordinare automaticamente tutti i numeri di linea appena cominciano ad essere un po' confusi. La versione che proponiamo è del genere "amatoriale", nel senso che il riordino viene realizzato escludendo i GOTO e i GOSUB (che vanno quindi modificati manualmente).

D'altra
parte, inserendo anche la rinumerazione dei salti il programma sarebbe diventato molto più lungo e complicato.
Eccoti il listato:

```
9000 REM RENUMBER
9010 STOP
9020 INPUT "INSERISCI IL NUMERO DI LINEA DI
       INIZIO RENUMBER",,NIR
9030 INPUT "INSERISCI IL NUOVO NUMERO DI
       LINEA INIZIALE",,NLI
9040 INPUT "INSERISCI L'INTERVALLO TRA LE
       LINEE",,ITL
9050 LET IM=PEEK(23635)+256*PEEK(23636)−1
9060 LET NX=PEEK(IM+1)*256+PEEK(IM+2)
9070 IF NX<NIR THEN GO TO 9120
9080 IF NX=9000 THEN GO TO 9140
9090 POKE (IM+1), INT(NLI/256)
9100 POKE (IM+2), NLI−256*INT(NLI/256)
9110 LET NLI=NLI+ITL
9120 LET IM=IM+4+PEEK(IM+3)+256*PEEK(IM+4)
9130 GO TO 9060
9140 LIST
```

Il programma va fatto girare con un RUN 9020 o con un GOTO 9020. Dopo di ciò appariranno le scritte in ingresso, che chiedono l'inserimento prima del numero di linea da cui iniziare la rinumerazione (per esempio 5), poi del nuovo numero di linea iniziale (per esempio 100) ed infine della distanza - cioè del passo - tra linea e linea. Naturalmente, è possibile effettuare anche più di una numerazione, per esempio per avere le prime 20 righe rinumerate con 1000, 1050, 1100, ecc.
Per comodità il programma può essere memorizzato su cassetta con il comando SAVE "RENUMBER" e caricato in memoria - quando occorre - tramite il comando MERGE "RENUMBER", che consente di inserire la routine in memoria senza cancellare il programma già presente.
Naturalmente, per usare il RENUMBER il programma da rinumerare non deve usare le linee da 9000 a 9140; inoltre vanno riservate al funzionamento della routine le variabili IM, NIR, NLI, ITL, NX.
A titolo di curiosità (ma non solo) riportiamo adesso una versione di RENUMBER scritta in linguaggio macchina; anch'essa svolge lo stesso lavoro del programma BASIC appena visto, cioè rinumerando semplicemente le linee

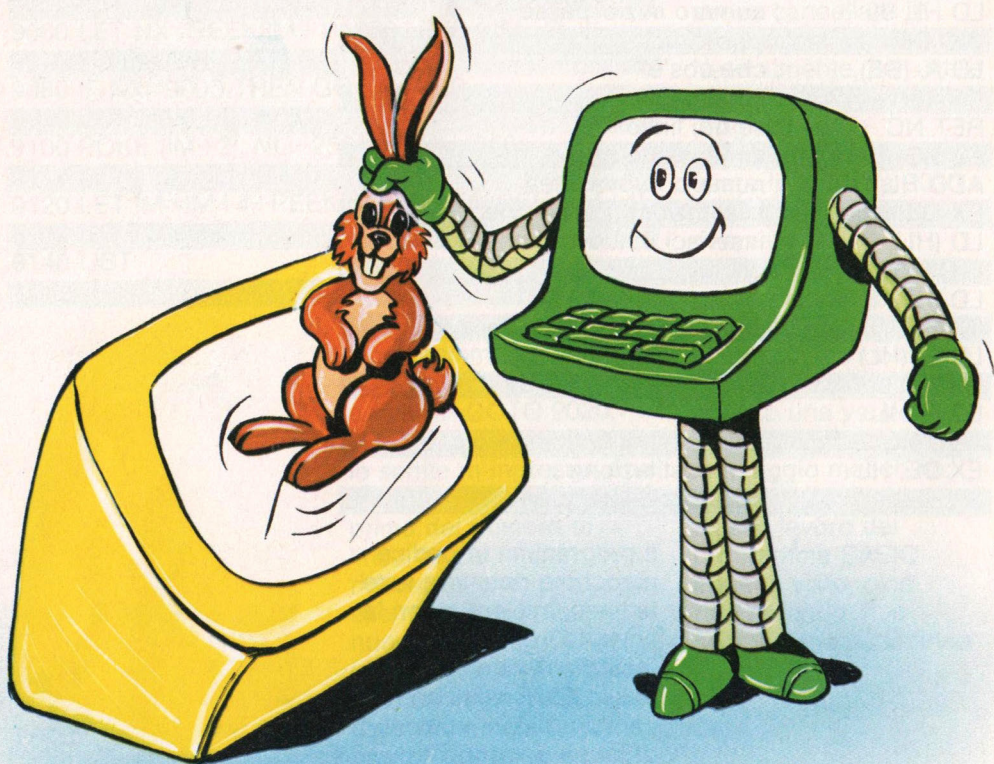
senza occuparsi dell'eventuale presenza di GOTO, GOSUB e RESTORE. La cosa spettacolare risulta tuttavia l'estrema rapidità con cui il compito viene portato a termine: la versione in linguaggio macchina risulta inoltre molto più corta della corrispondente versione BASIC, oltre che enormemente più veloce. Ecco il listato Assembler:

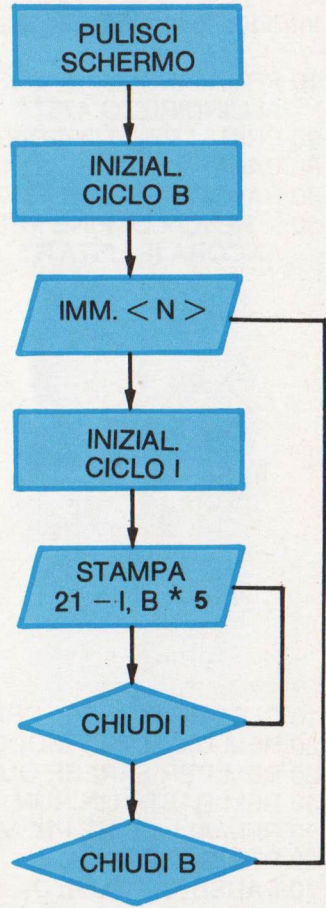
```
LD DE, 5CCAH   ; inizio del BASIC
LD HL, 90      ; numero inizio-passo
INC DE
LD A, (DE)     ; che cos'è?
CP 28H         ; fine linea?
RET NC         ; se no, finito
LD BC, 10      ; dimensione passo
ADD HL, BC     ; numero nuova linea
EX DE, HL      ; sostituzione temporanea
LD (HL), D     ; inserisci il nuovo numero di linea
INC HL
LD (HL), E
INC HL         ; prendi la lunghezza della linea
LD C, (HL)     ; mettila in BC
INC HL
LD B, HL
ADD HL, BC     ; posizione di fine linea
EX DE, HL      ; fine sostituzione
JR – 21        ; fai la prossima linea
```

Il programma in codice macchina rinumera tutte le linee, iniziando con la linea 100 e incrementando con passo 10. È possibile cambiare sia il numero di inizio, andando a modificare l'argomento della seconda linea (attualmente 90=100-10), che l'incremento, modificando con un altro numero il valore 10 dell'istruzione LD BC, 10. Per caricare la routine in memoria ti proponiamo adesso un breve programma BASIC, che si occupa di porre le istruzioni in linguaggio macchina in un'area dove la routine non possa essere sovrascritta.

PROGRAMMAZIONE

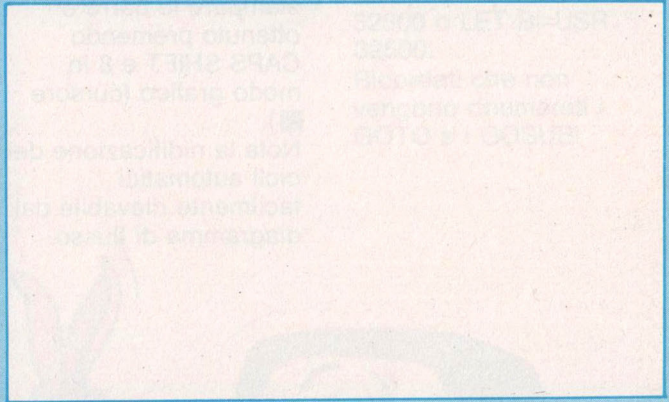
```
100 CLEAR 32499:LET A=32500
110 READ N:IF N=999 THEN STOP
120 POKE A,N
130 LET A=A+1
140 GO TO 110
150 DATA 17,202,92,33,90,0,19,26,254,40,208,1
160 DATA 10,0,9,235,114,35,115,35,78,35,70,9
170 DATA 235,24,235,999
```

Dopo aver inserito il programma in memoria basterà eseguirlo per ottenere la routine in memoria. Per far eseguire il renumber puoi impartire in modo immediato un comando USR, come PRINT USR 32500 o LET B=USR 32500.
Ricordati che non vengono rinumerati i GOTO e i GOSUB!

Rappresentazione grafica

Ecco un programma per generare un diagramma a barre.
Lo schermo visualizza fino a 5 elementi verticali in grado di rappresentare valori compresi tra 0 e 21. Il carattere utilizzato per stampare le barre è ottenuto premendo CAPS SHIFT e 8 in modo grafico (cursore G).
Nota la nidificazione dei cicli automatici facilmente rilevabile dal diagramma di flusso.

```
10 CLS
20 FOR B = 1 TO 5
30 INPUT "N ="; N
40 FOR I = 0 TO N
50 PRINT AT 21 – I, B * 5; "█"
60 NEXT I
70 NEXT B
```

VIDEOESERCIZI

Introduci i seguenti listati e attieniti a quanto detto nelle istruzioni REM.

```
10 PRINT "INDOVINA COSA FA LA ROUTINE DELLA ROM POSTA
   ALL'INDIRIZZO 4757"
20 PRINT : PRINT "PER SAPERLO PREMI UN TASTO."
30 PAUSE 1 : PAUSE 0 : CLS
40 RANDOMIZE USR 4757
50 > REM ALLA FINE PREMI LIST PER CONTROLLARE SE C'È
   ANCORA IL LISTATO
```

```
10 CLS : BEEP .5, 30 : PRINT "PREMI UN TASTO" : PAUSE 1 : PAUSE 0
20 REM UN ALTRO MODO PER EFFETTUARE UN COMANDO
30 REM PER SAPERE QUALE NON TI RESTA CHE PROVARLO
40 REM BASTA UN RUN
50 REM ALLA FINE PREMI UN TASTO
60 PRINT USR 6137
70 PAUSE 1 : PAUSE 0
```